Spring Framework

Spring is a powerful, lightweight framework used for application development. In broader terms, you can say that the Spring framework is a well-defined tool that supports several web applications using Java as a programming language.

Before the launching of the framework, the applications were developed using JEE standards. With these standards, we can deploy an application on any JEE application server. But, it had several problems, including:

1. Code became very complicated as application progressed.
2. Performance of the system got affected due to the heaviness of the applications.
3. The look-up problem of the component.

These problems were solved with the introduction of the Spring framework. The Spring framework became prominent in the market due to basic Spring framework features, which are its modularity. That is, it can be divided into different modules, each serving their own functionality.

Features of Spring Framework

Here are some most prominent features of Spring Framework:

1. Predefined templates
2. Easy to test
3. Loose coupling
4. Lightweight
5. Fast development
6. Powerful abstraction
7. Offers an array of resources
8. Declarative support
9. Offers comprehensive tools

History of Spring Framework

In October 2002, Rod Johnson, an Australian computer specialist, wrote a book titled Expert One-on-One J2EE Design and Development.

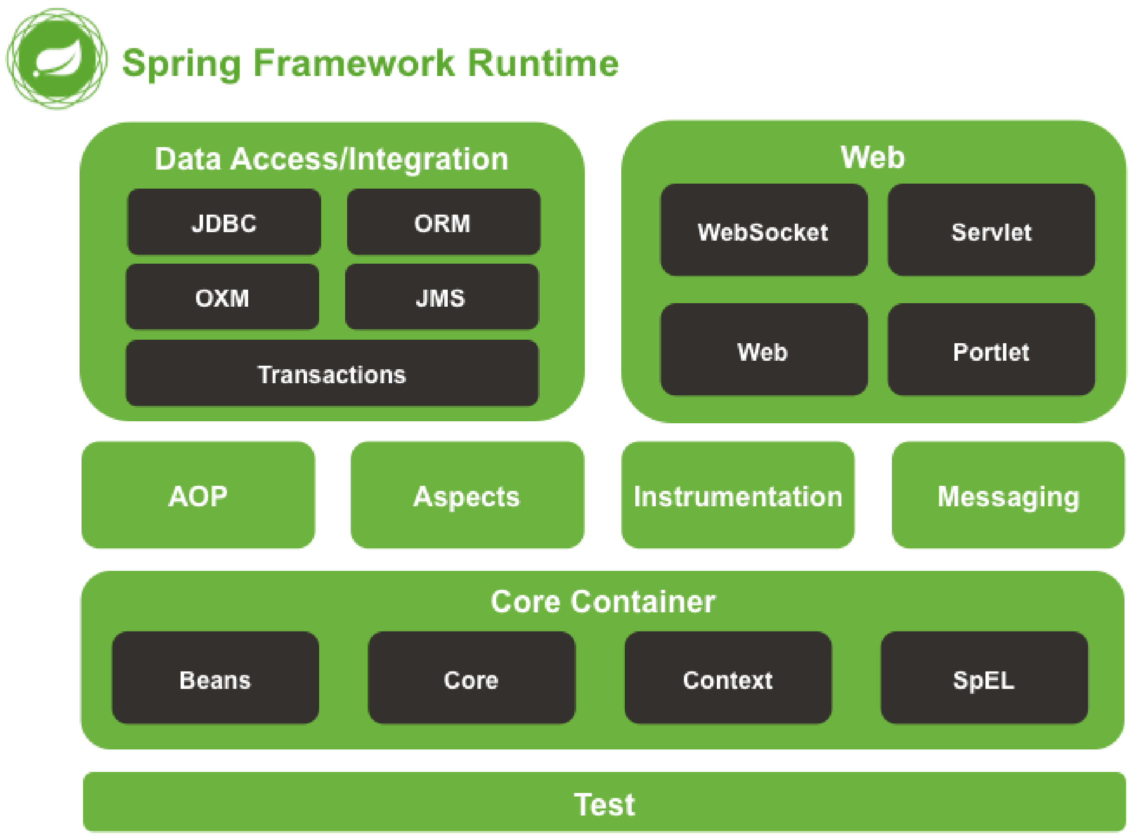
In this book he proposed a simpler solution based on ordinary Java classes (POJO) and dependency injection. He wrote over 30,000 lines of infrastructure code which included a number of reusable java interfaces and classes for developing the application. Around February 2003, Rod, Juergen and Yann started collaborating on the Spring project. The name “Spring” was given as it meant a fresh start after “Winter” of traditional J2EE.

Below is a timeline which shows the major version releases.

1. Spring framework was written by Rod Johnson and was first released in June 2002.
2. Spring last version release in March 2004
3. Spring 1.2.6 version release in 2006

4. Spring 2.0 version release in Oct 2006
5. Spring 2.5 version release in Nov 2007
6. Spring 3 version release in Dec 2009
7. Spring 3.1 version release in Dec 2011
8. Spring framework 4 version release in Dec 2013 with Java 8 support
9. Spring framework 4.2.0 version released on July 2015
10. Spring framework 4.2.1 version released in Sept 2015
11. Spring Framework 4.3 version released on 10 June 2016
12. Spring framework 5.0 version released on June 2017

Spring Framework Architecture



https://docs.spring.io/spring/docs/5.0.0.RC2/spring-framework-reference/overview.html

Inversion Of Control

1. Spring helps in the creation of loosely coupled applications because of Dependency Injection.

2. In Spring, objects define their associations (dependencies) and do not worry about how they will get those dependencies. It is the responsibility of Spring to provide the required dependencies for creating objects.
   - For example: Suppose we have an object Employee and it has a dependency on object Address. We would define a bean corresponding to Employee that will define its dependency on object Address.
   - When Spring tries to create an Employee object, it will see that Employee has a dependency on Address, so it will first create the Address object (dependent object) and then inject it into the Employee object.
3. Inversion of Control (IOC) and Dependency Injection (DI) are used interchangeably. IOC is achieved through DI. DI is the process of providing the dependencies and IOC is the end result of DI.
4. Inversion of Control can be achieved through various mechanisms such as: Strategy design pattern, Service Locator pattern, Factory pattern, and Dependency Injection (DI).
5. By DI, the responsibility of creating objects is shifted from our application code to the Spring container; **this phenomenon is called IOC**.
6. Dependency Injection can be done by setter injection or constructor injection.

Spring Containers

The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

- to instantiate the application class
- to configure the object
- to assemble the dependencies between the objects

There are two types of IoC containers. They are:

1. BeanFactory
2. ApplicationContext

Spring BeanFactory container

A **BeanFactory** is essentially an interface for an advanced factory capable of maintaining a registry of different beans and their dependencies. The BeanFactory enables you to read bean definitions and access them using the bean factory. When using just the BeanFactory you would create one and read in some bean definitions in the XML format as follows: